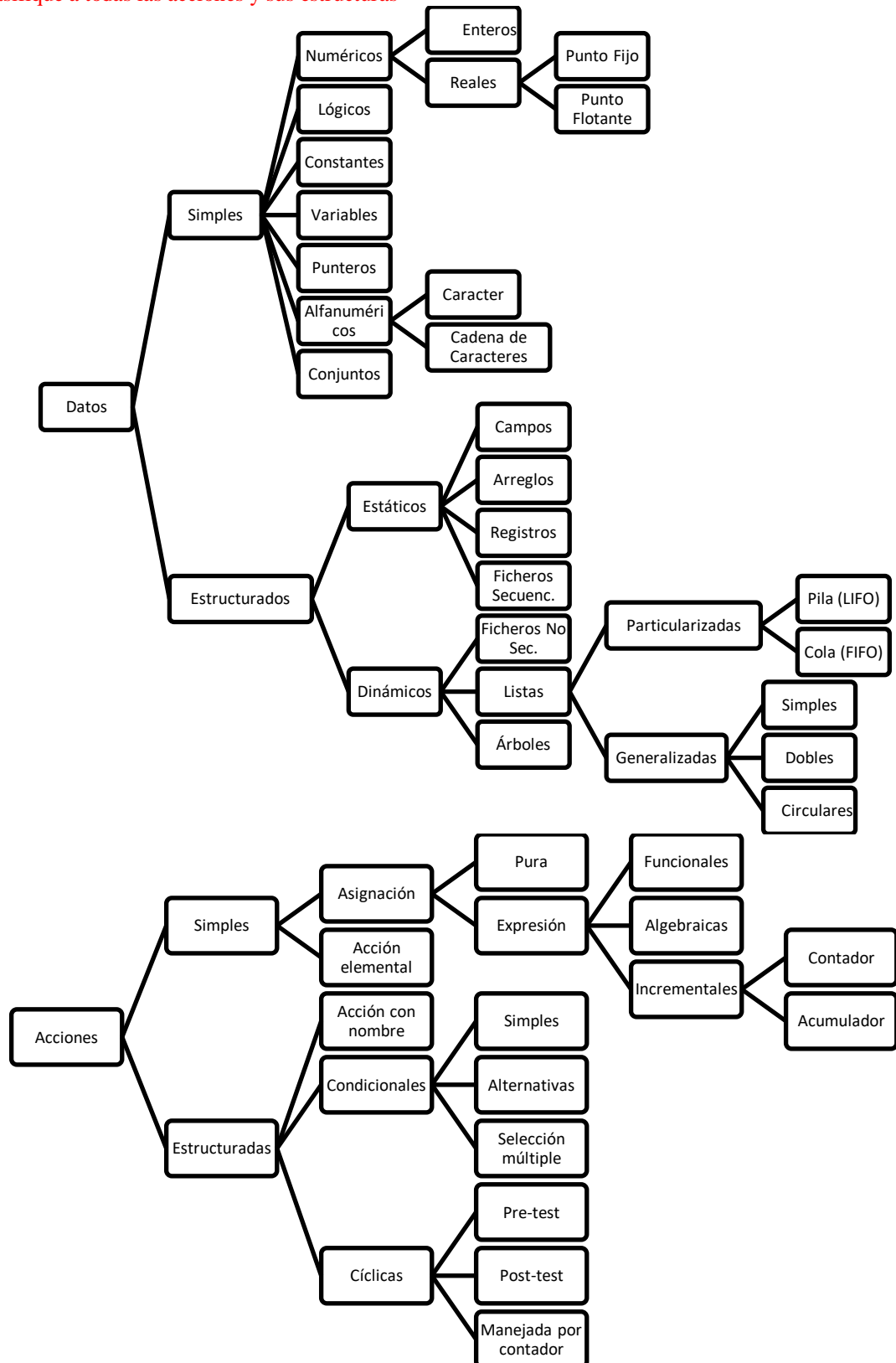


CLASIFICACIÓN DE DATOS Y ACCIONES

Clasifique en forma completa los datos y sus estructuras.

Clasifique a todas las acciones y sus estructuras



ESTRUCTURAS DE ACCIONES

Asignación pura	Receptor := Emisor (Emisor puede ser constante o variable)
Asignación funcional	Receptor := Función_Interna (Emisor) (Funciones internas: ABSO – SQRT – LN – LOG – EXP – TRUNC – REDOND – SQR – SIN – COS – TAN)
Asignación algebraica	Receptor := Emisor Operador Emisor (Operadores matemáticos: *, /, +, -, **, DIV, MOD)
Contador	Receptor := Receptor + 1
Acumulador	Acumulador := Acumulador + Variable
Acción con nombre	Acción NOMBRE es Acción 1; ... Acción n; Fin Acción
Condicional simple	Si CONDICIÓN entonces Acción 1; Fin Si
Condicional alternativo	Si CONDICIÓN entonces Acción 1; sino Acción 2; Fin Si
Condicional de selección múltiple	Según VARIABLE CONDICIONAL hacer valor 1: acción 1; ... valor n: acción n; otro: acción n+1; Fin Según
Ciclo Pre-Test (Rango: 0 a n)	Mientras CONDICIÓN hacer Acción 1; ... Acción n; Fin Mientras
Ciclo Post-Test (Rango: 1 a n)	Repetir Acción 1; ... Acción n; hasta que CONDICIÓN
Ciclo manejado por contador	Para VC:= VI hasta VF, I hacer Acción 1; ... Acción n; Fin Para

SECUENCIA

Explique el concepto de secuencia y subsecuencia y efectúe la clasificación de las mismas por todos los criterios.

Concepto: Un conjunto de elementos homogéneos está organizado en forma de secuencia si es posible definir las nociones siguientes:

- **Primer elemento de la secuencia:** un elemento del conjunto, llamado primero, se distingue de los demás. El acceso a este elemento permite el acceso a todos los demás elementos de la secuencia.
- **Relación de sucesión entre los elementos:** todo objeto de la secuencia (salvo el elemento final) precede a uno de los demás objetos (su sucesor).
- **Caracterización del fin de secuencia:** debe estar definido un indicador de fin de secuencia: caracteriza el elemento final, y en particular, permite detener la enumeración de la secuencia por observación de la característica del último elemento.

Las secuencias así definidas no autorizan el acceso a un elemento más que a través del elemento que le precede.

Clasificación de secuencias:

- Por su condición de fin:
 - **Puras:** El último elemento de la secuencia indica el fin de la misma y debe ser tratado como un elemento cualquiera. Se utiliza el ciclo Post-Test.
 - **Impuras:** Posee una marca de fin, es decir, un objeto extraño al final que no debe ser tratado como el resto de los elementos, se utiliza el ciclo Pre-Test.
- Por su cantidad de elementos:
 - **Definida:** Se conoce a priori la cantidad de elementos que posee.
 - **Indefinida:** No se conoce a priori la cantidad de elementos que posee.

Subsecuencias

Son secuencias de menor nivel que pertenecen a una secuencia mayor de la cual heredan sus características. Se pueden presentar de maneras enlazadas o jerarquizadas.

- **Subsecuencias enlazadas:** Son cadenas de Subsecuencias. Una subsecuencia termina cuando comienza la siguiente. Se encuentran enganchadas una a la otra.
- **Subsecuencias jerarquizadas:** Son Subsecuencias que van desde la de mayor importancia a la de menor importancia. Existe un elemento que sirve de puente. No hay continuidad ni encadenamiento.

RECETAS DE SECUENCIAS

Acción secuencia_pura es

Ambiente

S: Secuencia de caracteres

v: carácter

Algoritmo

ARRANCAR(S)

INICIALIZAR

Repetir

AVANZAR (S,v)

TRATAR_ELTO

hasta que ELTO_FINAL

TRATAR_FINAL

Fin acción

Acción secuencia_impura es

Ambiente

S: Secuencia de caracteres

v: carácter

Algoritmo

ARRANCAR(S)

INICIALIZAR

AVANZAR (S,v)

Mientras v <> marca hacer

TRATAR_ELTO

AVANZAR (S,v)

fin mientras

TRATAR_FINAL

Fin acción

Acción secuencia_definida es

Ambiente

S: secuencia de caracteres

v: carácter

i: entero

Algoritmo

ARRANCAR(S)

INICIALIZAR

Para i:= VI hasta VF, I hacer

AVANZAR(S,v)

TRATAR_ELTO

fin para

TRATAR_FINAL

Fin acción

Ejemplifique mediante una definición problemática.

Acción contar_alumnos_pura es Ambiente S: secuencia de caracteres v: carácter Cont: entero Algoritmo ARRANCAR(S) Cont := 0 Repetir AVANZAR (S,v) Cont := Cont + 1 Hasta que v = marca Escribir ("Hay", Cont, "alumnos") Fin acción	Acción contar_alumnos_impura es Ambiente S: secuencia de caracteres v: carácter Cont: entero Algoritmo ARRANCAR (S) AVANZAR (S,v) Cont := 0 Mientras v <> marca hacer Cont := Cont + 1 AVANZAR (S,v) Fin mientras Escribir ("Hay", Cont, "alumnos") Fin acción	Acción contar_alumnos_definida es Ambiente S: secuencia de caracteres v: carácter Cont, i: entero Algoritmo ARRANCAR (S) Cont := 0 Para i := 1 hasta tope hacer AVANZAR (S,v) Cont := Cont + 1 Fin para Escribir ("Hay", Cont, "alumnos") Fin acción
---	--	---

RECETAS DE SUBSECUENCIAS

<p>Acción Enlazada es</p> <p>Ambiente</p> <p>S: secuencia de caracteres</p> <p>v: caracter</p> <p>Algoritmo</p> <p>ARRANCAR(S)</p> <p>AVANZAR(S,v)</p> <p>INICIALIZAR</p> <p>Mientras v \diamond '.' hacer</p> <p>Mientras v = ' ' hacer</p> <p>TRATAR_BLANCO</p> <p>AVANZAR(S,v)</p> <p>Fin mientras</p> <p>Mientras v \diamond '.' ^ v \diamond ' ' hacer</p> <p>TRATAR_ELTO</p> <p>AVANZAR(S,v)</p> <p>Fin mientras</p> <p>Fin mientras</p> <p>TRATAR_FINAL</p> <p>Fin acción</p>	<p>Acción Jerárquica es</p> <p>Ambiente</p> <p>S: secuencia de caracteres</p> <p>v: caracter</p> <p>Algoritmo</p> <p>ARRANCAR(S)</p> <p>AVANZAR(S,v)</p> <p>INICIALIZAR</p> <p>Mientras v \diamond marca_hoja hacer</p> <p>Mientras v \diamond marca_párrafo hacer</p> <p>Mientras v \diamond '.' hacer</p> <p>TRATAR_CHARACTER</p> <p>AVANZAR(S,v)</p> <p>Fin mientras</p> <p>TRATAR_FINAL_ORACIÓN</p> <p>AVANZAR(S,v)</p> <p>Fin mientras</p> <p>TRATAR_FINAL_PÁRRAFO</p> <p>AVANZAR(S,v)</p> <p>Fin mientras</p> <p>TRATAR_FINAL_HOJA</p> <p>AVANZAR(S,v)</p> <p>Fin acción</p>
---	---

REGISTRO

Defina el concepto de registro y efectúe su clasificación y ejemplificación de los tipos de registros

Concepto: Estructura de datos que contiene información referida a una entidad y está compuesta de un número fijo de componentes llamados “campos”, donde cada uno de ellos viene definido con un nombre y un tipo. Es una estructura estática, compleja y se almacena en memoria externa. Los campos que contiene pueden ser continentes y contenidos.

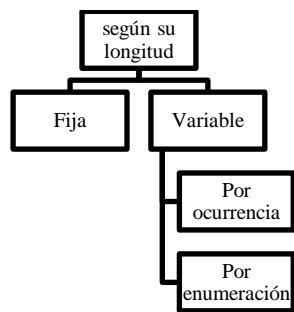
Campo continente: Contiene a otros campos. Es complejo. Tiene la estructura segmentada.

Campo contenido: Contenido por otro campo. Es simple. No tiene segmentaciones. Requiere que se le defina tipo y tamaño.

REPRESENTACIÓN DE REGISTROS

Lineal, gráfica o esquemática	Jerárquica o arbórea	Literaria						
<p>Celdas continuas, donde cada dato tiene su cuadro.</p> <p>Fecha</p> <table border="1"> <tr> <td>Día</td><td>Mes</td><td>Año</td></tr> <tr> <td></td><td></td><td></td></tr> </table>	Día	Mes	Año				<p>Cada nodo representa una entidad y se relacionan mediante líneas de flujos.</p> <pre> graph TD Fecha[Fecha] --> Día[Día] Fecha --> Mes[Mes] Fecha --> Año[Año] </pre>	<p>Cada elemento ocupa un renglón de código. La subordinación está dada por el uso de sangrías.</p> <p>REG = registro</p> <p>Campo1: tipo1</p> <p>...</p> <p>Campo n: tipo n</p> <p>Fin registro</p>
Día	Mes	Año						

TIPOS DE REGISTROS (CLASIFICACIÓN Y EJEMPLIFICACIÓN)



Longitud fija

ALUMNO = Registro
 AyN: AN(30);
 Legajo: N(5);
 Fin registro

Longitud variable por ocurrencia

UTN = Registro
 Legajo: N(5);
 CANT: 0..42;
 Materia: Registro ocurre según CANT
 Cod_mat: N(3);
 Fecha: N(8);
 Fin registro
 Fin registro

Longitud variable por enumeración

MOVA = Registro
 CLAVE = Registro
 CodOP: N(1)
 Acción: N(2)
 Fin registro
 CodMov: 'A'
 Fin registro

MOVB = Registro
 CLAVE = Registro
 CodOP: N(1)
 Acción: N(2)
 Fin registro
 CodMov: 'B'
 Fin registro

MOVC = Registro
 CLAVE = Registro
 CodOP: N(1)
 Acción: N(2)
 Fin registro
 CodMov: 'C'
 Fin registro

BAJAS = Registro
 CLAVE = Registro
 CodOP: N(1)
 Acción: N(2)
 Fin registro
 CodMov: ('D', 'E')
 Fin registro

ArchMov: Archivo de MOVA, MOVB, MOVC, BAJAS.

ARCHIVO

Describe el concepto de archivo. Defina organización y acceso, clasifique las distintas categorías y sus vinculaciones.

Concepto: Es una estructura de datos que colecciona datos que son todos del mismo tipo y que se encuentran almacenados en una memoria externa permanente. La estructura de un fichero se caracteriza porque su cardinalidad es infinita. Por esta razón no hace falta dar la longitud del fichero, pero su cardinalidad es potencialmente infinita por la capacidad de la memoria externa. A todo fichero hay que darle nombre y tipo.

Organización: Es la manera en cómo van a ser almacenados los datos dentro de un fichero. La organización es permanente, ya que una vez definida no puede cambiarse. La forma de almacenamiento nace y muere con el archivo.

Acceso: Es la manera en la que se van a leer (recuperar) los registros de un fichero.

Clasificación:

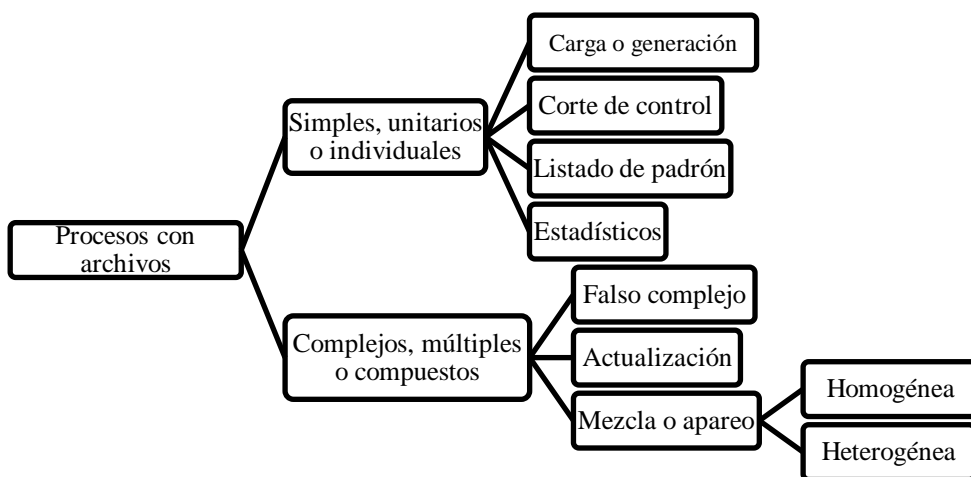
Por su **organización**:

- Secuenciales. → Acceso secuencial.
- No secuenciales:
 - Relativos. → Acceso secuencial y directo
 - Indexados. → Acceso secuencial, directo y dinámico

Por su **acceso**:

- Secuencial.
- Directo.
- Dinámico o mixto

PROCESOS CON ARCHIVOS Enuncie un cuadro sinóptico con todos los procesos simple y complejos de archivos.



1 - FICHeros SECUENCIALES

TIPO	PROCESO	CARACTERÍSTICAS
INDIVIDUALES Un proceso es individual cuando existe un único Fichero de Entrada y 1 o ningún Fichero de Salida.	Genérico	Es el proceso de carga o generación, tiene como objetivo crear un archivo consistente y de ser posible, congruente grueso.
	Emisión	Tiene como objetivo la salida impresa de datos. Son listadores cuando emiten listados como se ingresan sin más que títulos. Se considera padrón cuando los datos ingresados están ordenados y se emiten además totales finales.
	Estadísticos	Recorrido del archivo para contabilizar elementos, utilizando una tabla (memoria interna) y al finalizar emitir un cuadro de resumen.
	Corte De Control	Son padrones, pero poseen totales parciales. Es requisito obligatorio que el archivo de entrada esté ordenado por clave compleja.
MÚLTIPLES Un proceso es múltiple cuando existen 2 o más Ficheros de Entrada y 1 o más Ficheros de Salida.	Mezcla	Ficheros de entrada: por lo menos dos. Ficheros de salida: uno (resultado de la combinación de los dos de entrada).
	Actualización Secuencial Por Lotes	Cero o más registros del fichero movimiento por cada registro del maestro
	Actualización Secuencial Unitaria	Cero o un (como máximo) registro del fichero movimiento por cada registro del maestro.

Diferencia entre corte y padrón

Si se solicita que un archivo esté ordenado por cierto campo, entonces tenemos un padrón. Un proceso Corte de Control es un padrón con totales parciales. Es necesario, para hacer el corte, que el archivo esté ordenado por clave compleja. El total final se obtiene con el padrón. Los totales parciales, con el corte de control.

Concepto de clave

La clave es un dato dentro del registro que permite identificar unívocamente un dato, dado que no existen dos datos iguales en el archivo. Una clave es simple cuando es un campo contenido. Una clave es completa cuando es campo continente.

¿Qué condiciones se tienen que dar para que un archivo sea correcto?

Debe cumplir dos condiciones:

Ser consistente: Que la información que se esté guardando coincida con el formato usado en la definición. Leer un dato desde un archivo o pantalla y verificar que cumpla con las características de la salida. Dicho proceso se debe realizar con la carga de datos.

Ser congruente: Se verifica la consistencia de cada elemento consigo mismo. Se compara a un elemento con algún otro. Un archivo es consistente grueso cuando el dato es correcto. Se comparan los datos de un mismo registro y se observa si se cumplen. Compara, por ejemplo, que un valor esté dentro del rango dado.

Explique en qué consiste la técnica de apareo y sus distintos ciclos. Ventajas e inconvenientes de cada uno.

El **apareo o mezcla** es el proceso en el cual intervienen al los menos dos ficheros de entrada, que deben ser combinados para obtener uno de salida. Existe una clave que sirve como conector entre los archivos de entrada. Esta mezcla puede ser directa o indirecta.

En la **mezcla directa** todos los ficheros intervinientes tienen el mismo formato de registro, y el número de registros que poseerá el fichero de salida es la sumatoria de los registros de los ficheros de entrada.

En la **mezcla indirecta**, de todos los ficheros intervinientes hay uno que se considera de mayor prioridad y es el que maneja el ciclo de mezcla, los restantes pueden ser de la misma forma del anterior o distintos y la salida adopta la forma del fichero de mayor prioridad, una mezcla entre las formas intervinientes, no pudiéndose establecer una relación numérica en función de sus elementos.

Existen dos **ciclos** que nos permiten realizar la mezcla.

CICLOS INCLUYENTES: Todos los archivos de entrada son tratados en el mismo ciclo.	CICLOS EXCLUYENTES: Se tratan los archivos comunes en el ciclo y los no comunes fuera de él.
Mientras NO FDA(A1) o NO FDA(A2) hacer Procesamiento_archivos; Fin mientras	Mientras NO FDA(A1) y NO FDA(A2) hacer Procesamiento_archivos; Fin mientras MIENTRAS NO FDA(A1) hacer Procesamiento_loquefalta_archivo1; Fin mientras MIENTRAS NO FDA(A2) hacer Procesamiento_loquefalta_archivo2; Fin mientras

Comparar actualización secuencial con actualización indexada. Mejoras y desventajas.

ACTUALIZACIÓN SECUENCIAL	ACTUALIZACIÓN INDEXADA
El resguardo es automático. (Backup automático)	El resguardo es provocado.
No puedo modificar el archivo maestro original. (Diferido)	Sí puedo modificar el archivo maestro original. (In Situ)

Diferencias físicas y lógicas entre archivos secuenciales e indexados. Clasifique las diferentes categorías y sus vinculaciones.

ARCHIVOS SECUENCIALES	ARCHIVOS INDEXADOS
Poseen acceso secuencial. Para acceder al último elemento se debe acceder a todos los anteriores.	Poseen acceso secuencial y directo a cualquier registro.
Son estructuras estáticas que una vez creadas no se pueden ampliar. No hace falta dar su longitud.	Son estructuras dinámicas que pueden crecer o reducirse. No hace falta dar su longitud.
Los registros se almacenan secuencialmente en memoria con adyacencia física.	Los registros se almacenan en memoria con adyacencia lógica.
Para insertar un registro, hay que crear un nuevo archivo	

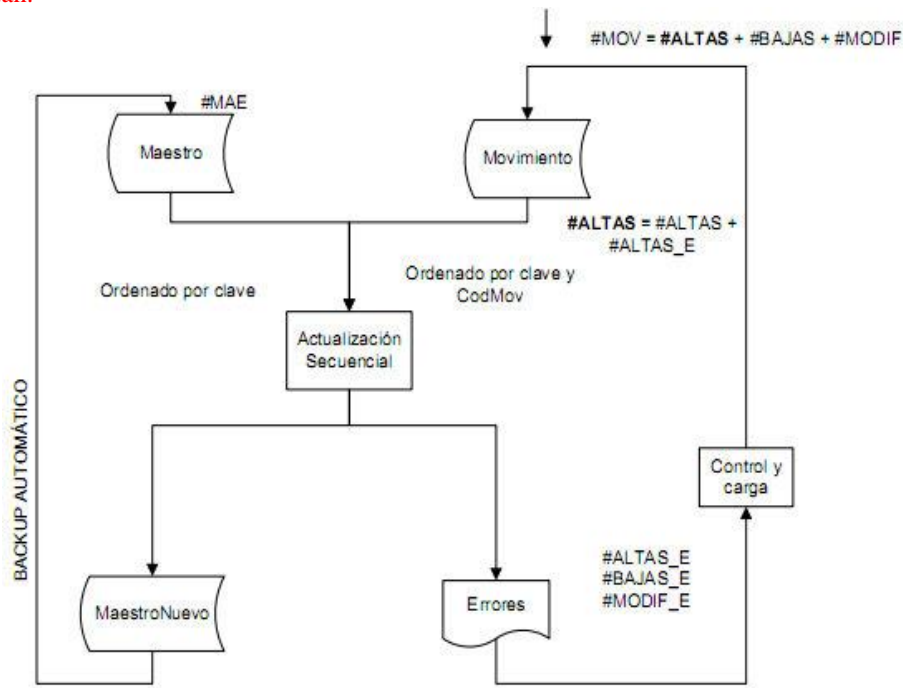
¿Cuál es la causa para que un proceso de actualización sea considerado interactivo?

La actualización interactiva IN SITU se llama así porque se cuenta con una terminal, ya que no existe un archivo de movimiento. Como es una terminal, aquí se encuentra un usuario inteligente (con poder de decidir). Quien corrige los errores es el usuario. Este proceso termina cuando la terminal termina de funcionar.

Diferencias entre procesos IN SITU y procesos diferidos.

PROCESOS DIFERIDOS	PROCESOS IN SITU
Los movimientos están en un archivo.	No existe un archivo de movimientos. Se cuenta con una terminal.
Los errores se corrigen cada cierto período de tiempo.	Los errores los corrige el usuario inteligente.
Es un proceso seguro, ya que puede ser cuantificado.	

Esquematice el proceso de actualización secuencial con todos sus componentes y explicitando sobre los mismos, los controles que efectúan.

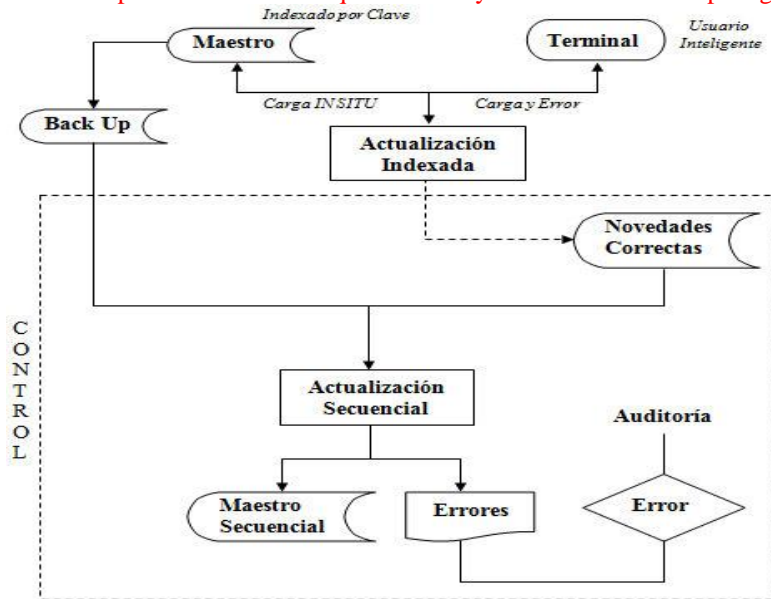


Referencias

- #MAE: número de registros existentes en el archivo maestro.
- #MOV : número de registros existentes en el archivo movimiento, compuesto por:
 $\#ALTAS + \#BAJAS + \#MODIF$
- #MN: número de registros grabados en el nuevo archivo maestro, compuesto por:
 $\#MAE - \#BAJAS + \#ALTAS$
- #ALTAS: cantidad de altas correctas.
- #BAJAS: cantidad de bajas correctas.
- #MODIF: cantidad de modificaciones correctas.
- #ALTAS_E: cantidad de altas erróneas
- #BAJAS_E: cantidad de bajas erróneas.
- #MODIF_E: cantidad de modificaciones erróneas.

Características: Lento, seguro, diferido, batch.

Esquematice el proceso de actualización indexada con todos sus componentes e indique las características principales del mismo. Explícite los controles que efectúan y/o deberían efectuar para garantizar la seguridad de los datos.



Características: Interactivo, rápido, inseguro, In Situ, Back Up

ARREGLO

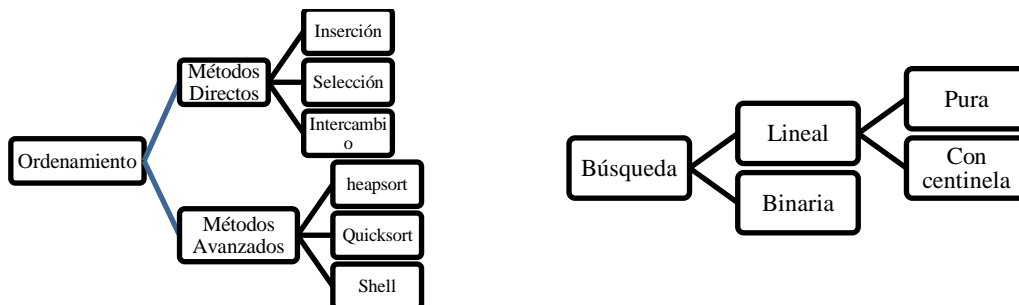
Describa el concepto de arreglo en su totalidad.

Es una estructura de datos estática en la que se almacena una colección de datos del mismo tipo, y se caracteriza por:

- Almacenar los elementos del arreglo en posiciones físicas contiguas de memoria.
- Tener un único nombre de variable que representa a todos los elementos y cada uno de estos se diferencian por un índice.
- Tener acceso directo a los elementos individuales.
- La necesidad de definir su tamaño antes de su uso.

Existen tres tipos de arreglos: unidimensionales, bidimensionales y multidimensionales.

Clasifique todos los métodos de ordenamiento y búsqueda de arreglos. Elija uno y explique detalladamente y luego algoritmice.



Búsqueda lineal pura: Este tipo de búsqueda consiste en recorrer el vector en su totalidad e ir comparando cada elemento del mismo con el elemento buscado para verificar su existencia. Como se puede observar, este tipo de búsqueda no finaliza al encontrar la primera ocurrencia, sino que apunta a la multitud de éxitos, es decir, se encarga de tratar el elemento cada vez que lo encuentra. En caso de no ser encontrado el elemento se debe informar la no existencia del mismo.

```

Leer (X);
exito:= Falso;
Para i:=1 hasta n hacer
    Si A[i] = X entonces
        exito:= Verdadero;
        Escribir ("El elemento se ha encontrado en la posición: ", i);
    Fin si
Fin para
Si No exito entonces
    Escribir ("El elemento buscado no ha sido encontrado");
Fin si

```

Búsqueda lineal con centinela: En este tipo de búsqueda el recorrido del vector es parcial siempre y cuando el elemento buscado no se situé en la última posición del mismo o no exista el mismo. Como podemos observar la búsqueda finaliza cuando encuentra la primera ocurrencia o ante primer valor mayor que el elemento buscado (en el caso de que arreglo se encuentre ordenados). El número de comparaciones como mínimo será el número de elementos que haya antes de encontrar el elemento buscado. Para su implementación en arreglos desordenados se debe preguntar, en la cabecera del ciclo de búsqueda, si el vector en la posición i es distinto (" \neq ") al elemento buscado. En cambio, en los arreglos ordenados, se debe colocar el símbolo de mayor ">" o menor "<" según en qué orden se encuentre el arreglo.

```

Leer (X);
i:=1;
Mientras (i < n) y (A[i] <math>\neq X</math>) hacer
    i:= i+1;
Fin mientras
Si A[i] = X entonces
    Escribir ("El elemento encontrado en posición: ", i);
sino
    Escribir ("El elemento no encontrado");
Fin si

```

Búsqueda binaria: Este tipo de búsqueda se basa en el principio "divide y vencerás" y es sólo aplicable a arreglos ordenados y básicamente consiste en comparar el elemento central del vector con el valor buscado. Si estos no son iguales se reduce el intervalo de búsqueda a la mitad derecha o izquierda, según sea el valor buscado mayor o menor que el elemento central respectivamente. Este proceso se repite reiteradamente y termina cuando se encuentra el elemento o bien cuando se anula el intervalo de búsqueda. El número de comparaciones máximo es $\log_2 N$ (siendo N el tamaño del vector).

```

Leer (X);
Inf:=1; Sup:=N;
medio:=(Inf+Sup) div 2;
Mientras (Inf < Sup) y (A[medio] <math>\neq X</math>) hacer
    Si A[medio] < X entonces
        Inf:= medio+1;
    sino
        Sup:= medio-1;
    Fin si
    medio:=(Inf+Sup) div 2;
Fin mientras
Si A[medio] = X entonces
    Escribir ("El elemento se ha encontrado en la posición: ", medio);
sino
    Escribir ("El elemento buscado no ha sido encontrado");
Fin si

```

Ordenamiento por inserción directa: En este tipo de ordenamiento el arreglo está dividido conceptualmente en dos secuencias, una secuencia destino ($1 \dots i$) y otra origen ($j=i+1 \dots n$). Partiendo con $i=1$ e incrementando i de uno en uno, se toma el elemento j de la secuencia de origen y se transfiere a la secuencia de destino, insertándolo en el sitio adecuado. El proceso de encontrar el sitio adecuado consiste en comparar el elemento j -ésimo con su antecesor y determinar el menor de los dos, en caso de ser el j -ésimo elemento el menor (en caso de ordenar de menor a mayor) se procede al intercambio de lugar de los elementos. Este proceso continúa hasta que:

1. Se encuentra un elemento menor que la del elemento j -ésimo.
2. Se alcanza el extremo izquierdo de la secuencia de destino (es decir el principio del vector).

```

Para i:=1 hasta A N-1 hacer
    J:=i+1;
    Mientras (j>1) y (A[j] < A[j-1]) hacer
        AUX:=A[j]; A[j]:= A[j-1];
        A[j-1]:=AUX; j:=j-1;
    Fin mientras
Fin para

```

Ordenamiento por Selección: Éste método se basa en los siguientes principios:

1. Encontrar el elemento menor.
2. Intercambiarlo con el primero, luego estas operaciones se repiten con los elementos $n-1$, $n-2$, etc.

En primer lugar se recorre el arreglo desde el primer al último elemento para encontrar el menor, una vez hallado el mismo se lo intercambia con el primer elemento pasando, de ésta manera, a ser el primer elemento del vector ordenado. Ésta operación se irá repitiendo partiendo desde el primer elemento a la derecha del vector ordenado. Es posible afirmar que en general, el método de selección directa es preferible al de inserción directa, aunque en los casos en los que las claves están inicialmente ordenadas o casi ordenadas, éste último puede llegar a ser algo más rápido.

```

Para i:=1 hasta A N-1 hacer
    min:=i;
    Para j:=i+1 hasta N hacer
        Si A[j]<A[min] entonces
            min:=j;
        Fin si
    Fin para
    aux:=A[min];
    A[min]:=A[i];
    A[i]:=aux;
Fin para

```

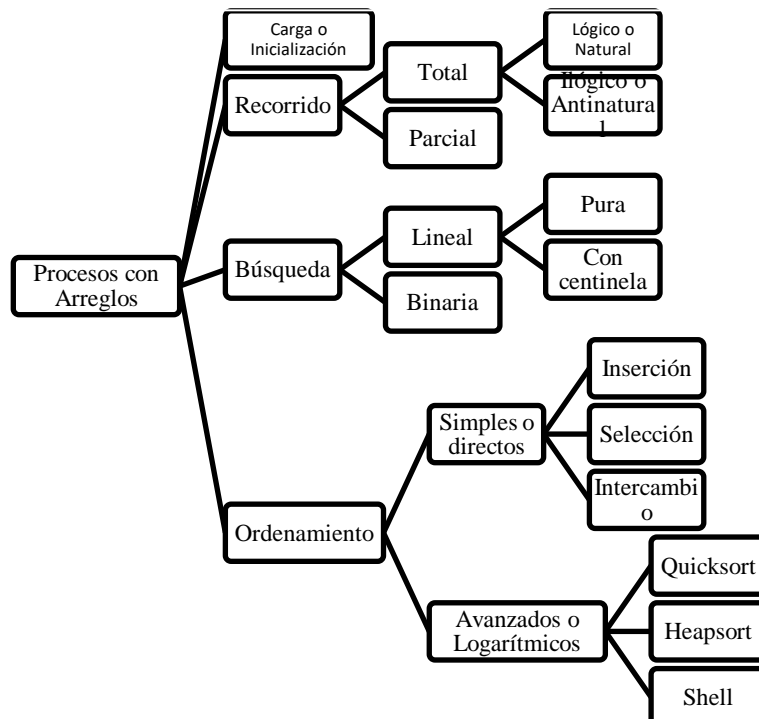
Intercambio (Burbuja): Este algoritmo se basa en el principio de comparar e intercambiar pares de elementos adyacentes hasta que todos estén ordenados. Se lo conoce como el método de la “Burbuja” porque si se mira el arreglo como si estuviera en posición vertical, en vez de horizontal, y se imagina que es un tanque de agua, cada burbuja (elemento) asciende con un peso acorde a su valor. En cada pasada, una burbuja asciende hasta el nivel de peso que le corresponde. Se comparan el primer y segundo elemento del arreglo y si se encuentran desordenados se los intercambia, inmediatamente se comparan el segundo con el tercero y así sucesivamente hasta que no se produzcan cambios en el arreglo, es decir, cuando se encuentre ordenado.

```

Bandera:= Falso
Mientras No Bandera hacer
    Bandera := Verdadero
    Para i:= 1 a n-1 hacer
        Si (A[i] < A [i+1]) entonces
            x := A[i]
            A[i] := A[i+1]
            A[i+1] := x
            Bandera := Falso
        Fin si
    Fin para
Fin mientras

```

Clasifique los procesos con arreglos.



RECURSIVIDAD

Explique y describa en qué consiste la técnica recursiva, en qué casos se aplica, ejemplifique cada uno. Compare los procesos recursivos con los iterativos.

Técnica recursiva: Se dice que algo es recursivo si forma parte de sí mismo o se define en función de sí mismo. El poder de la recursividad reside en la posibilidad de definir un número infinito de objetos mediando un enunciado finito. De igual manera, un número infinito de operaciones de cálculo, puede describirse mediante un programa recursivo finito. La facultad de variar su tamaño es la propiedad característica de las estructuras recursivas que claramente las distingue de las estructuras fundamentales.

La recursividad es un proceso rápido pero que ocupa mucho espacio en memoria. Un caso recursivo puede plantearse siempre de forma iterativa, pero el recíproco no es siempre válido.

La recursividad puede ser a nivel de definición de datos y/o de acciones.

Datos recursivos: Es aquella donde el objeto se define en función de sí mismo. Se define un objeto infinito, mediante una expresión finita y unitaria. Ejemplo: listas simplemente encadenada.

P: Puntero a ALUMNO

ALUMNO = Registro

Legajo: N(5)

NyA: AN(30)

PROX: Puntero a ALUMNO

Fin registro

Acción recursiva: Es aquella en la que un subprograma se llama a sí mismo para resolver un problema el cual se resuelve mediante la descomposición del mismo en subproblemas más simples de un tipo similar. Al ir reduciendo progresivamente la complejidad del problema a resolver, llegará un momento en que habrá un caso (o varios) tan simple que pueda resolverse directamente sin necesidad de hacer otra llamada recursiva. En esa situación diremos que estamos ante un **caso base** de la recursividad. Ha de existir al menos un caso base para evitar que la recurrencia sea infinita.

- Planteo recursivo
 - Caso base: Aquel que puede ser resuelto de forma iterativa
 - Caso recursivo: Es una versión menor del programa, que no puede resolverse de manera iterativa. Se lo puede descomponer hasta llegar al caso base.

Tipos de recursividad: Existen dos tipos de recursividad:

- **Directa:** Un subprograma se llama a sí mismo una o más veces directamente.
Acción POTENCIA (X,N) es
Si N = 0 entonces
 POTENCIA := 1
sino
 POTENCIA := POTENCIA (X, N-1)
Fin si
Fin acción
- **Indirecta:** Un subprograma A llama a otro subprograma B y este, a su vez, llama al subprograma A.

Función PAR (N: Entero): Booleano

Si N = 0 entonces

 PAR := Verdadero

sino

 PAR := IMPAR (N-1)

Fin si

Fin función

Función IMPAR (N: Entero): Booleano

Si N = 0 entonces

 IMPAR := Falso

sino

 IMPAR := PAR (N-1)

Fin si

Fin función

Ejemplo: Si hacemos la llamada IMPAR(3) hace las siguientes llamadas:

- PAR(2)
- IMPAR(1)
- PAR(0)

Devuelve 1 (Cierto). Por lo tanto 3 es número impar.

FUNCIÓN RECURSIVA	PROCEDIMIENTO RECURSIVO
Acción FACTORIAL (N: entero) es Si $N = 0$ entonces FACTORIAL := 1 sino FACTORIAL := $N * \text{FACTORIAL}(N-1)$ Fin si Fin acción (Devuelve sólo un resultado)	Acción INVERSO (N: entero) es Escribir($N \text{ MOD } 10$) Si $N \geq 10$ entonces INVERSO ($N \text{ DIV } 10$) Fin si Fin acción (Puede devolver uno o varios resultados)

Obtener el número inverso de un número N (siendo $N \geq 0$), el cual se genera invirtiendo las cifras. Por ejem: El inverso de 198 es 891.

Si pudiera elegir entre una solución iterativa y una recursiva, ¿Cuál elegiría y por qué?

Escriba el planteo formal indicando sus casos. Y escriba el procedimiento recursivo.

¿Qué tipos de recursividad conoce y cuál de ellas es la que usa en el algoritmo escrito?

Subacción INVERTIR (N: entero) es

Escribir ($N \text{ MOD } 10$)

Si $(N \text{ DIV } 10 > 0)$ entonces

 INVERTIR ($N \text{ DIV } 10$)

Fin si

Fin subacción

Elegiría una solución recursiva porque el problema en sí está planteado en forma recursiva. Para obtener el inverso de un número, debo invertir sus cifras, por lo tanto estaría haciendo una recursividad en la función "INVERTIR". El tipo de recursividad utilizado en este ejercicio es la Recursividad Directa.

PROCESOS RECURSIVOS	PEROCESOS ITERATIVOS
La repetición de las tareas se controla desde adentro.	La repetición de tareas se controla desde afuera.
Se ejecuta el conjunto de acciones y antes de finalizar, se evalúa si se ha llegado a la condición de salida, sino se continúa adelantando para ejecutar un nuevo conjunto de acciones.	Se ejecuta un conjunto de acciones, en forma completa, se verifica la condición de fin y si se necesita se vuelve a ejecutar el conjunto entero de acciones.
Suele ser lento y ocupa mayor espacio en memoria, pero es más entendible por ser el mecanismo de razonamiento que usa el ser humano.	Normalmente es más sencillo y más eficiente, pues usa menos recursos.
Todo proceso recursivo puede ser iterativo.	No todo proceso iterativo puede ser recursivo.

ÁRBOLES

Explique concepto de Árbol Binario, ejemplifique; indique formas de recorrido. Algoritmice los recorridos. Efectúe una representación mediante una ecuación matemática.

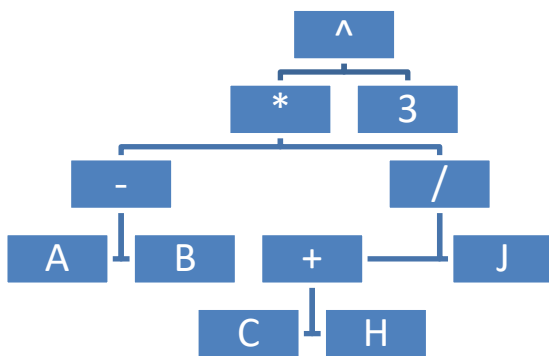
Árboles: son estructuras no lineales de datos que organizan sus elementos, denominados *nodos*, formando jerarquías. Cada nodo del árbol contiene la información que le es propia y las direcciones de otros nodos de menor jerarquía que él con los cuales se relaciona. Cada nodo puede relacionarse con cero, uno, o más elementos. Generalmente se utilizan estas estructuras para representar relaciones.

Árbol binario: Se caracteriza por ser un árbol en el que cada nodo no puede tener más de dos hijos o descendientes. Es decir, es un conjunto de nodos que es o bien el conjunto vacío o un conjunto que consta de un **nodo raíz** enlazado a dos árboles binarios disjuntos denominados **subárbol izquierdo** y **subárbol derecho**. Cada uno de estos subárboles es, a su vez, un árbol binario.

Tipos de recorrido: Existen tres tipos de recorrido:

PRE-ORDEN	EN ORDEN	POST-ORDEN
1) Visitar la raíz 2) Recorrer subárbol izquierdo 3) Recorrer subárbol derecho	1) Recorrer subárbol izquierdo 2) Visitar la raíz 3) Recorrer subárbol derecho	1) Recorrer subárbol izquierdo 2) Recorrer subárbol derecho 3) Visitar la raíz
Ambiente P: Puntero a Nodo Nodo = Registro Dato: entero izq, der: P Fin registro Algoritmo Acción pre_orden es Si $p \neq \text{nil}$ entonces Escribir ($P^{\wedge}.\text{Dato}$) Pre_orden ($P^{\wedge}.\text{izq}$) Pre_orden ($P^{\wedge}.\text{der}$) Fin si Fin acción	Ambiente P: Puntero a Nodo Nodo = Registro Dato: entero izq, der: P Fin registro Algoritmo Acción en_orden es Si $p \neq \text{nil}$ entonces En_orden ($P^{\wedge}.\text{izq}$) Escribir ($P^{\wedge}.\text{Dato}$) En_orden ($P^{\wedge}.\text{der}$) Fin si Fin acción	Ambiente P: Puntero a Nodo Nodo = Registro Dato: entero izq, der: P Fin registro Algoritmo Acción post_orden es Si $p \neq \text{nil}$ entonces Post_orden ($P^{\wedge}.\text{izq}$) Post_orden ($P^{\wedge}.\text{der}$) Escribir ($P^{\wedge}.\text{Dato}$) Fin si Fin acción

Dado la siguiente fórmula: $X = [(A-B) * [(C+H) / J]]^3$ grafique el árbol binario que lo represente. Efectúe sobre este ejemplo los tres tipos de recorrido.



Recorrido EN ORDEN: $(A-B) * [(C+H) / J]]^3$
 Recorrido PRE ORDEN: $^ * - A B / + C H J 3$
 Recorrido POST ORDEN: $A B - C H + J / * 3 ^$

COMPLEJIDAD

Defina cuales son los criterios para medir el rendimiento de un algoritmo ¿Cómo se mide la eficiencia temporal? ¿Cuál es el principio de invarianza?

Los **criterios para medir el rendimiento** de un algoritmo se centran principalmente en su simplicidad y en el uso eficiente de los recursos. La sencillez es una característica muy interesante a la hora de diseñar un algoritmo, ya que facilita su verificación, el estudio de su eficiencia y su mantenimiento. Es por esto que muchas veces se considera más importante la simplicidad y legibilidad del código frente a otras alternativas más difíciles de entender y eficientes del algoritmo. Respecto al uso eficiente de los recursos, éste suele medirse en función de dos parámetros: la memoria que utiliza y el tiempo que tarda en ejecutarse. Los parámetros que maneja este último criterio nos van a servir además para comparar algoritmos entre sí, permitiendo determinar el más adecuado de entre varios que solucionan un mismo problema.

El tiempo de ejecución de un algoritmo va a depender de diversos factores como son: los datos de entrada que le suministremos, la calidad del código generado por el compilador para crear el programa objeto, la naturaleza y rapidez de las instrucciones máquina del procesador concreto que ejecute el programa, y de la complejidad propia del algoritmo.

Existen dos maneras distintas de **medir la eficiencia temporal**:

1. La primera forma consiste en obtener una función que acote (por arriba o por abajo) el tiempo de ejecución del algoritmo para unos valores de entrada dados. Esto nos proporciona una medida teórica (a priori) que nos ofrece estimaciones del comportamiento de los algoritmos de forma independiente del ordenador en donde serán implementados y sin necesidad de ejecutarlos.
2. La segunda forma consistente en medir el tiempo de ejecución del algoritmo para unos valores de entrada dados y en un ordenador concreto. Esta técnica nos ofrece una medida real (a posteriori) del comportamiento del algoritmo.

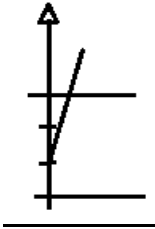
Principio de Invarianza: Dado un algoritmo y dos implementaciones suyas I_1 e I_2 , que tardan $T_1(n)$ y $T_2(n)$ segundos respectivamente, el Principio de Invarianza afirma que existe una constante real $c > 0$ y un número natural n_0 tales que para todo $n \geq n_0$ se verifica que $T_1(n) \leq cT_2(n)$.

En pocas palabras, el tiempo de ejecución de dos implementaciones distintas de un algoritmo dado no va a diferir más que en una constante multiplicativa.

Para la siguiente subacción, calcular las asíntotas:

SUBACCIÓN	Euclides (m,n: CARDINAL):	OE
CARDINAL;		
Var temp: CARDINAL;		
MIENTRAS m>0 HACER	(*1*)	2
Temp := m;	(*2*)	1
m := n MOD m;	(*3*)	2
n := temp;	(*4*)	1
END;	(*5*)	
RETURN n;	(*6*)	
END Euclides;		

Cálculo de asíntotas

Peor Caso: Mayor cantidad de acciones $\sum_{m=n}^0 (2 + 1 + 2 + 1) + 2$ $\sum_{m=n}^0 6 + 2$ $\sum_{m=1}^n 6 + 2$ $N(6) + 2$ $6n + 2$	Mejor Caso: Menor cantidad de acciones 2 (Por lo que hay dentro del Mientras)  Orden de complejidad: Lineal
--	---